1 Первое знакомство с JavaScript

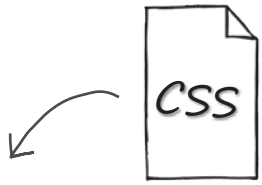
В незнакомых водах

JavaScript открывает фантастические возможности. JavaScript, **основной язык программирования** Всемирной паутины, позволяет **добавлять поведение** к веб-страницам. Забудьте о сухих, скучных, статичных страницах, которые просто занимают место на экране, — с JavaScript вы будете взаимодействовать с пользователями, реагировать на события, получать и использовать данные из интернета, выводить графику на страницах... и многое, многое другое. При хорошем знании JavaScript вы сможете дальше программировать **совершенно новое** поведение своих пользователей.

И вы будете в хорошей компании. JavaScript — не только один из **самых популярных** языков программирования, он еще и **поддерживается** всеми современными браузерами; более того, появились встроенные реализации JavaScript, существующие отдельно от браузеров. А впрочем, хватит разговоров. Пора браться за дело!

Как работает JavaScript

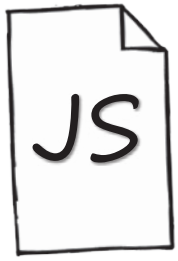
Вы освоили создание структуры, контента, макета и стиля веб-страниц. Не пора ли добавить к ним поведение? В наше время страница, на которую можно только смотреть, никому не интересна. Хорошие страницы должны быть *динамическими* и *интерактивными* и по-новому взаимодействовать с пользователями. Именно для этого и нужен JavaScript. Для начала давайте посмотрим, какое место JavaScript занимает в *экосистеме веб-страниц*:



Вы уже знаете, что язык HTML (Hypertext Markup Language) используется для определения **контента** страниц, а также их **структуры**: абзацев, заголовков и разделов.

Браузер

Вы знаете, что для отображения HTML-разметки используется технология каскадных таблиц стилей CSS (Cascading Style Sheets) — цвета, шрифты, границы, макет страницы... CSS определяет **стиль** оформления, отделенный от структуры страницы.

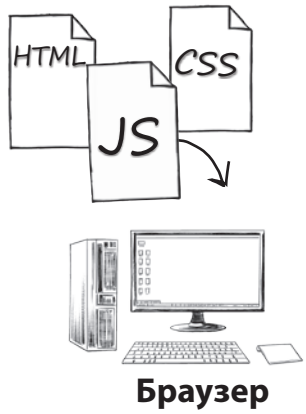


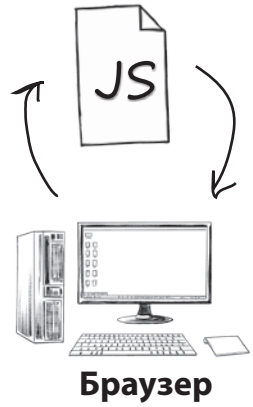
Итак, знакомьтесь: JavaScript, родственник HTML и CSS. JavaScript умеет определять **поведение** веб-страниц. Нужно среагировать, когда пользователь нажимает кнопку «Распродажа заканчивается через 30 секунд!»? Быстро проверить содержимое текстового поля с важными данными? Загрузить данные с сайта NASA и вывести их на странице? А может, хотите поиграть? На помощь приходит JavaScript. Вы сможете добавить к странице программный код, который будет выполнять вычисления, реагировать на события, рисовать, передавать данные, сигнализировать, обновлять информацию, изменять... Словом, любые динамические операции — всё это умеет делать JavaScript.

Как пишется код JavaScript

JavaScript занимает *особое* место в мире программирования. Как появляется типичная классическая программа? Вы пишете код, компилируете его, проводите компоновку и устанавливаете на компьютер. Язык JavaScript куда более гибок и динамичен. Программист включает код JavaScript прямо в страницу, а потом загружает ее в браузер. Далее браузер сам сделает все необходимое для выполнения написанного кода. Давайте повнимательнее присмотримся к тому, как работает эта схема.

```
<html>
<head>
<title>Ice Cream</title>
<script>
 let x = 49;
</script>
</head>
<body>
<h1>Ice Cream Flavors</h1>
<h2><em>49 flavors</em></h2>
<p>All your favorite
flavors!</p>
</body>
</html>
```





1 Написание кода

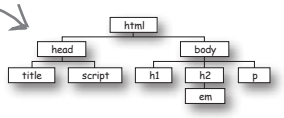
Страница создается как обычно: с контентом HTML и стилевым оформлением CSS. На страницу добавляется код JavaScript. Как вы вскоре увидите, по аналогии с HTML и CSS все компоненты можно разместить в одном файле или же выделить код JavaScript в отдельный файл, который включается в страницу.

Вскоре мы разберемся, какой способ лучше...

2 Загрузка

Откройте страницу в браузере. Обнаружив в странице код JavaScript, браузер сразу начинает разбирать его и готовить к выполнению. Как и в случае с HTML и CSS, если браузер обнаруживает ошибки в JavaScript, он пытается продолжить чтение JavaScript, HTML и CSS. Браузер старается избежать ситуации, в которой пользователь не сможет увидеть запрошенную страницу.

На будущее: браузер строит «объектную модель» страницы HTML, которая может использоваться кодом JavaScript. Просто запомните этот факт, мы еще к нему вернемся...



3 Выполнение

Браузер начинает выполнять код сразу, как только встречает его, и продолжает выполнять на всем протяжении жизненного цикла страницы. Некоторые части кода выполняются только один раз (до перезагрузки страницы); другие выполняются каждый раз, когда пользователь что-то делает, например, щелкает на кнопке или перемещает мышь. И то и другое будет рассмотрено в книге.

Как включить код JavaScript в страницу

Начнем сначала: чтобы продвинуться в изучении JavaScript, необходимо знать, как включить код в страницу. Как же это делается? Конечно, при помощи элемента <script>!

Давайте возьмем скучную старую веб-страницу и определим для нее динамическое поведение в элементе <script>. Пока не нужно задумываться над смыслом того, что мы включаем в элемент <script>, — сейчас важнее понять, как вообще работает JavaScript.

Стандартный заголовок HTML doctype, элементы <html> и <head>.

Элемент <body> выглядит вполне традиционно.

В раздел <head> страницы добавляется элемент <script>.

В нем записывается фрагмент кода JavaScript.

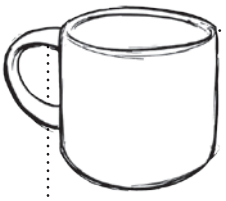
```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Just a Generic Page</title>
    <script>
      setTimeout(wakeUpUser, 5000);
      function wakeUpUser() {
         alert("Are you going to stare at this boring page forever?");
      }
    </script>
  </head>
  <body>
       <h1>Just a generic heading</h1>
       <p>Not a lot to read about here. I'm just an obligatory paragraph living
in an example in a JavaScript book. I'm looking for something to make my life more
exciting.</p>
  </body>
</html>
```

Еще раз: сейчас для нас не важно, как работает этот код. И все же... Посмотрите на него и попробуйте предположить, что происходит в каждой из строк.

Тест-драйв

Введите код страницы и сохраните его в файле с именем «behavior.html». Теперь загрузите страницу в браузере (перетащите файл в окно браузера или воспользуйтесь командой Файл > Открыть). Что делает наш код?

Подсказка: чтобы это понять, нужно подождать пять секунд.



Расслабьтесь

Только не волнуйтесь. Никто не ждет, что вы сразу начнете понимать JavaScript так, словно знаете его с детства. Сейчас достаточно представлять, на что похож JavaScript.

Впрочем, совсем расслабляться тоже не стоит: нужно, чтобы ваш мозг заработал в полную силу. Помните код с предыдущей страницы? Давайте попробуем предположить, что происходит в каждой строке:

Вероятно, мы создаем код, который можно будет использовать в других местах, и присваиваем ему имя «wakeUpUser»?

```
setTimeout(wakeUpUser, 5000);
function wakeUpUser() {
    alert("Are you going to stare at this boring page forever?");
}
```

Здесь каким-то образом отсчитывается 5 секунд? Подсказка: 1000 миллисекунд = 1 секунда.

Тут все понятно: выводится сообщение для пользователя.

Часто Задаваемые Вопросы

В: Язык JavaScript как-то связан с Java?

О: Только по названию. Язык JavaScript создавался на пике популярности Java, и разработчики JavaScript удачно воспользовались этим обстоятельством. Оба языка заимствуют некоторые элементы синтаксиса из языков семейства C, но в остальном они имеют мало общего.

В: Я слышал, что JavaScript называют «игрушечным языком». Это правда?

О: На первых порах JavaScript не отличался мощью, но затем его значимость возросла, и на расширение возможностей JavaScript были направлены значительные ресурсы (в том числе и умы лучших специалистов). Но знаете что? Еще до того, как JavaScript стал таким быстрым, он был замечательным. И как вы вскоре убедитесь, с ним можно сделать много интересного.

В: Значит, JavaScript — лучший способ создания динамических страниц?

О: JavaScript — основной язык программирования для браузера. Существуют такие разновидности JavaScript, как TypeScript, но код TypeScript преобразуется в код JavaScript перед выполнением в браузере. С современными сверхбыстрыми средами JavaScript и хитроумными API JavaScript так и остается стандартом программирования для браузера.

В: Мой друг работает с JavaScript в музыкальном приложении... по крайней мере он так говорит. Это возможно?

О: Да! JavaScript — универсальный язык сценарного программирования и проникает во многие приложения, от графических редакторов до музыкальных программ, и даже в область серверного программирования. Скорее всего, ваши вложения в изучение JavaScript оправдаются и за пределами веб-разработки.

В: Вы говорите, что многие другие языки компилируются. Что это такое и почему этого нет в JavaScript?

О: В традиционных языках программирования — C, C++ или Java — код компилируется перед выполнением. В процессе компиляции код преобразуется в представление, понятное для машины (и обычно оптимизированное по скорости выполнения). Сценарные языки обычно *интерпретируются*, то есть браузер выполняет каждую строку JavaScript сразу же, как только встретит ее. Для сценарных языков производительность во время выполнения не так важна; они в большей степени ориентированы на построение прототипов, интерактивное программирование с максимальной гибкостью. С ранними версиями JavaScript так и было, по этой причине еще много лет скорость выполнения кода была довольно посредственной. Однако существовал промежуточный вариант: интерпретируемый язык, который компилируется «на ходу». Именно он был выбран разработчиками браузеров для современных реализаций JavaScript. По сути, в JavaScript вы пользуетесь всеми удобствами сценарных языков в сочетании с быстродействием компилируемого языка. Кстати говоря, в этой книге часто встречаются слова «интерпретировать», «вычислять» и «выполнять». Они могут различаться по смыслу в разных контекстах, но для наших целей фактически эквивалентны.

JavaScript, ты проделал длинный путь...

JavaScript 1.0

Возможно, вы не помните Netscape, но он был первым *настоящим* разработчиком браузеров. В середине 1990-х на рынке шла яростная борьба (особенно со стороны Microsoft), и добавление новых интересных возможностей в браузер было исключительно важным делом.

Для этого компания Netscape создала язык сценариев, который позволял любому желающему включить сценарный код в страницу. Так появился LiveScript — язык, разработанный для удовлетворения этой потребности в краткосрочной перспективе. Скорее всего, вы никогда не слышали о LiveScript, потому что в то же время Sun Microsystems представила язык Java, и ее акции взлетели до заоблачного уровня. Почему бы не воспользоваться чужим успехом? Так LiveScript стал JavaScript.

У этих языков нет ничего общего? Ну и что...

А что же Microsoft? Вскоре за Netscape она создала собственный язык сценариев, который назвали... JScript. Он был как-то подозрительно похож на JavaScript. Так начались войны браузеров и непростые времена для разработчиков.

JavaScript 2015

После всей путаницы с JavaScript наконец-то была достигнута определенность. На свет появился ECMAScript — официальное определение языка для всех реализаций JavaScript (в браузерах и без них).

В 2015 году JavaScript наконец-то достиг зрелости и заслужил признание профессиональных разработчиков. Это объяснялось как появлением надежного стандарта, так и серьезным вкладом со стороны разработчиков браузеров (таких, как Google), которые вывели JavaScript на профессиональную сцену с Google Maps и другими сложными приложениями, построенными на платформе браузера.

Благодаря этому изменению позиции многие лучшие умы из области программирования занялись совершенствованием интерпретаторов JavaScript и повышением их эффективности на стадии выполнения. После выхода JavaScript 2015, критического обновления языка, модель выпуска переключилась с официальной нумерации версий на ежегодные обновления.

JavaScript 2024

В наши дни JavaScript занимает положение *основного* языка веб-разработки. Благодаря предварительной компиляции интерпретация кода JavaScript в веб-страницах выполняется с молниеносной быстротой. Средства проверки синтаксиса, редакторы кода с поддержкой синтаксиса и мощные средства отладки на базе браузеров сделали веб-программирование профессиональной областью. JavaScript — один из самых популярных языков в мире, который был реализован на самых разнообразных платформах: встроенных системах, веб-серверах и даже музыкальных приложениях.

В настоящее время язык достиг зрелости. Обновления невелики и быстро интегрируются с браузерами. Многочисленные библиотеки и фреймворки JavaScript означают, что на JavaScript можно сделать практически все, что можно сделать на любом другом языке. Несмотря на свою странную и неоднозначную историю, JavaScript добился успеха!

Посмотрите, как легко пишется код JavaScript

Вы еще не знаете JavaScript, но наверняка сможете хотя бы в общих чертах предположить, как работает его код. Взгляните на каждую строку и попробуйте догадаться, что она делает. Запишите свои ответы. Мы приводим один ответ, чтобы вам было проще взяться за дело. А если задача окажется непосильной, ответы приведены на следующей странице.

```
let price = 28.99;
let discount = 10;
let total =
      price - (price * (discount / 100));
if (total > 25) {
   freeShipping();
}

let count = 10;
while (count > 0) {
   juggle();
   count = count - 1;
}

const dog = {name: "Rover", weight: 35};
if (dog.weight > 30) {
    alert("WOOF WOOF");
} else {
    alert("woof woof");
}

let circleRadius = 20;
let circleArea =
   Math.PI * (circleRadius * circleRadius);
```

Создать переменную price и присвоить ей значение 28.99..

→ Решение на с. 42

Посмотрите, как легко пишется код JavaScript

Решение

См. с. 41

Вы еще не знаете JavaScript, но наверняка сможете хотя бы в общих чертах предположить, как работает его код. Взгляните на каждую строку и попробуйте догадаться, что она делает. Ниже приведены наши ответы.

`let price = 28.99;`	*Создать переменную price и присвоить ей значение 28.99.*
`let discount = 10;`	*Создать переменную discount и присвоить ей значение 10.*
`let total =` `  price - (price * (discount / 100));`	*Вычислить цену со скидкой и присвоить переменной total.*
`if (total > 25) {`	*Сравнить переменную total и 25. Если переменная больше...*
`  freeShipping();`	*...выполнить фрагмент кода freeShipping.*
`}`	*Конец команды if.*
`let count = 10;`	*Создать переменную count и присвоить ей значение 10.*
`while (count > 0) {`	*Пока значение count остается больше 0...*
`  juggle();`	*...что-то сделать, а потом...*
`  count = count - 1;`	*...уменьшить count на 1.*
`}`	*Конец цикла while.*
`const dog = {name: "Rover", weight: 35};`	*Создать константу dog с атрибутами name и weight.*
`if (dog.weight > 30) {`	*Если атрибут weight больше 30...*
`  alert("WOOF WOOF");`	*...вывести на веб-странице сообщение «WOOF WOOF».*
`} else {`	*А если нет...*
`  alert("woof woof");`	*...вывести на веб-странице сообщение «woof woof».*
`}`	*Конец команды if/else.*
`let circleRadius = 20;`	*Создать переменную circleRadius и присвоить ей значение 20.*
`let circleArea =`	*Создать переменную с именем circleArea...*
`  Math.PI * (circleRadius * circleRadius);`	*...и присвоить ей результат выражения (1256.6370614359173).*

Страницы, построенные на базе HTML и CSS, могут хорошо выглядеть. Но изучив JavaScript, вы сможете строить принципиально новые разновидности страниц.

А заодно и подзаработать!

Более того, их будет правильнее рассматривать не как обычные страницы, а как приложения!

Что вы говорите? «Конечно, я это отлично знаю, иначе зачем бы мне читать эту книгу?» Вообще-то мы хотели воспользоваться возможностью и немного поговорить об изучении JavaScript. Если у вас уже имеется опыт работы на каком-нибудь языке программирования или языке сценариев, то вы примерно представляете, что вас ждет. Но если до сегодняшнего дня вы ограничивались HTML и CSS, знайте: при изучении языка программирования вас ждет нечто принципиально новое.

В HTML и CSS в основном выполняются декларативные операции. Допустим, вы объявляете, что некоторый текст абзаца или все элементы класса «sale» должны быть окрашены в красный цвет. JavaScript добавляет в страницу новое *поведение*, а для этого необходимо описать вычисления. Вам понадобятся средства для описания разнообразных операций: «вычислить счет игрока, сложив количество баллов», или «повторить следующее действие десять раз», или «когда пользователь нажмет эту кнопку, воспроизвести такой-то звуковой сигнал», или даже «выйти на улицу, узнать температуру воздуха и разместить ее на этой странице».

Язык, необходимый для решения таких задач, сильно отличается от HTML и CSS. Давайте посмотрим, чем именно...

Как создаются команды

При создании контента HTML вы обычно размечаете текст, определяя его структуру. Для этого в текст добавляются элементы, атрибуты и значения:

```
<h1 class="drink">Mocha Caffe Latte</h1>
<p>Espresso, steamed milk, and chocolate syrup,
just the way you like it.</p>
```

При работе с HTML мы размечаем текст для определения структуры, например: «Так, здесь у нас будет крупный заголовок, а за ним следует абзац обычного текста».

С CSS дело обстоит немного иначе. Разработчик пишет набор **правил**; каждое правило выбирает элементы страницы, а затем задает для этих элементов набор стилей:

```
h1.drink {
    color: brown;
}
p {
    font-family: sans-serif;
}
```

Для CSS пишутся правила с селекторами (например, h1.drink и p), определяющими, к каким частям разметки HTML применяется данный стиль.

Допустим, все заголовки drink выводятся коричневым цветом...

...а абзацы выводятся шрифтом без засечек.

Код JavaScript состоит из **команд**. Каждая команда описывает небольшую часть выполняемой операции, а весь набор команд определяет поведение страницы:

```
let age = 25;
let name = "Owen";

if (age > 14) {
    alert("Sorry this page is for kids only!");
} else {
    alert("Welcome " + name + "!");
}
```

Набор команд

Каждая команда выполняет небольшую часть работы, например объявляет переменные, в которых будут храниться используемые значения.

Мы создаем переменную для хранения возраста (25). Нам понадобится и переменная для имени «Owen».

Значение переменной может использоваться для принятия решений. Возраст пользователя больше 14?

Если больше — сообщаем, что пользователь слишком стар для этой страницы.

А если нет — приветствуем пользователя по имени (впрочем, в нашем примере Оуэну 25 лет, так что сообщение не выводится).

Переменные и значения

Вероятно, вы заметили, что в программах JavaScript обычно используются переменные. Переменные предназначены для хранения значений. Каких? Рассмотрим несколько примеров:

```
let winners = 2;
```

Команда объявляет переменную с именем winners и присваивает ей числовое значение 2.

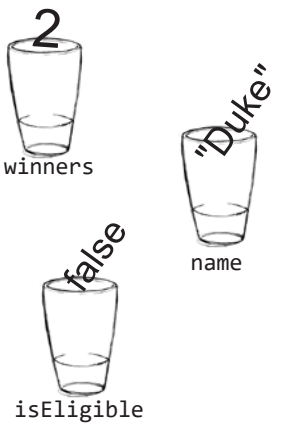
```
let name = "Duke";
```

Переменной присваивается последовательность символов (такая последовательность называется строкой).

```
let isEligible = false;
```

Переменной isEligible присваивается значение false. Переменные, принимающие два значения, true и false (истина/ложь), называются логическими (или булевыми).

В честь математика Джорджа Буля.



Кроме чисел, строк и логических значений переменные могут хранить и другие данные. Вскоре мы доберемся и до них, но независимо от вида данных все переменные создаются по одним правилам. Присмотримся повнимательнее к объявлению переменной:

Объявление переменной всегда начинается с ключевого слова let.

Даже если JavaScript не пожалуется, что вы пропустили let, переменные всегда должны объявляться. Вскоре вы поймете почему...

Далее указывается имя переменной.

```
let winners = 2;
```

Команда присваивания всегда завершается точкой с запятой.

Наконец, при желании можно указать начальное значение переменной; для этого ставится знак равенства, за которым следует значение.

Мы говорим «при желании», потому что, строго говоря, вы можете создать переменную без начального значения и присвоить его позднее. Для этого достаточно убрать из команды присваивание:

```
let losers;
```

Если переменная объявляется без знака равенства и значения, значит, вы просто собираетесь ее как-то использовать в будущем.